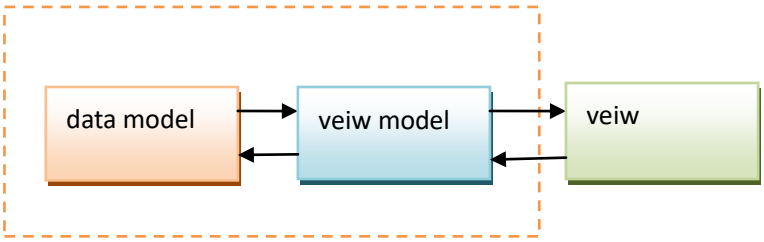
konckout+bootstarp 组件化 MVVM 的实现

背景

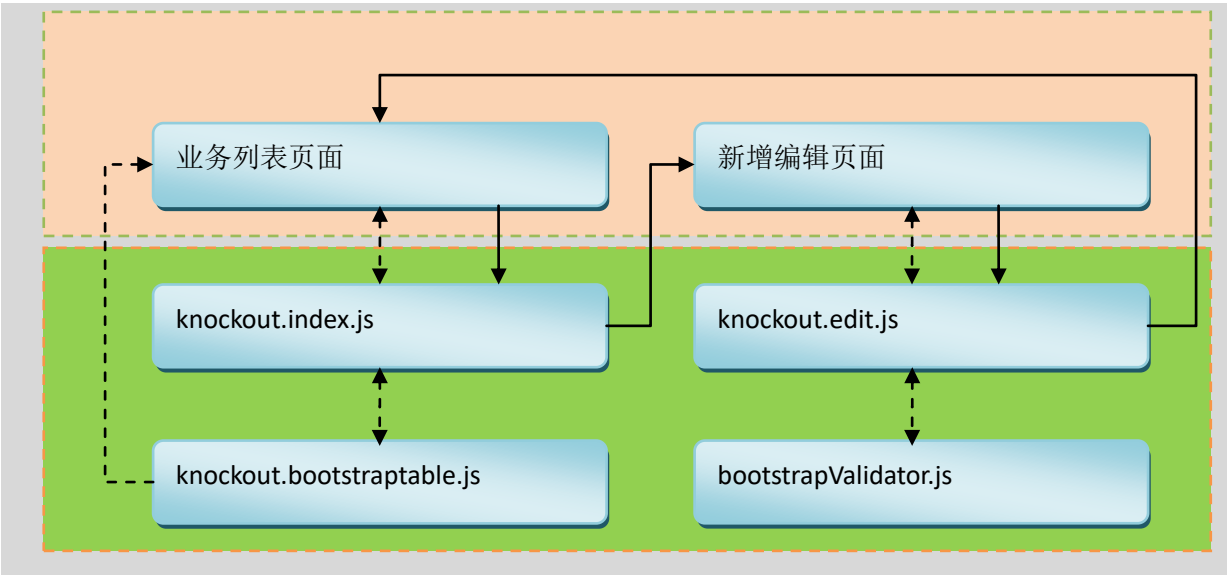
私有云 web 后台更新。在没有前端的前提下，开发先行。开发需要减少前端开发耗时，更需要满足后续的更新扩展。因此需要能便捷开发和独立开发。开发人员可以专注与业务逻辑和数据的开发(ViewModel)。设计人员可以专注于界面(View)的设计。这样可以减少当前开发耗时以及后续的页面更新耗时。

设计思路

mvvm 的详解示意图：

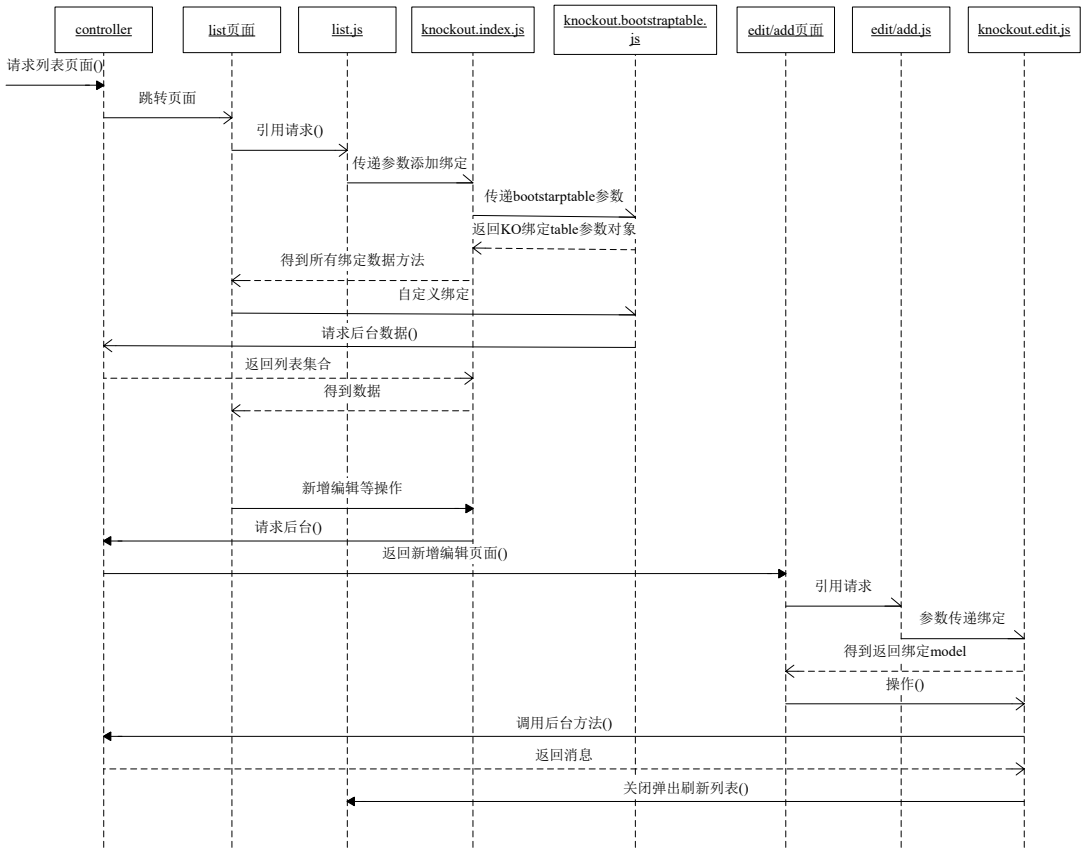


代码组件化封装示意图：



注释：实线代表操作，虚线代表内部调用
绿色背景是组件，橙色是 veiw 层

代码逻辑时序图：



代码说明

- **knockout.bootstraptable.js**

1、代码包含初始的 bootstarptable 对象默认参数。

```
    //向ko里面新增一个bootstrapTableViewModel方法
    ko.bootstrapTableViewModel = function(options) {
        var that = this;
        //bootstarp默认参数设置
        this.defaultSetting = {
            toolbar: '#toolbar',                //工具按钮用哪个容器
            queryParamsType:"",                 //数据查询l类型
            pagination: true,                   //是否显示分页（*）
            sidePagination: "server",           //分页方式：client客户端分页，server服务端分页（*）
            pageNumber: 1,                      //初始化加载第一页，默认第一页
            pageSize: 15,                       //每页的记录行数（*）
            pageList: [15, 25, 50, 100],        //可供选择的每页的行数（*）
            method: 'post',                     //请求方式
            search: false,                      //是否显示表格搜索，此搜索是客户端搜索，不会进服务端，所以，
            strictSearch: true,                 //数据严格搜索,true启用全匹配搜索，否则为模糊搜索
            showColumns: true,                  //是否显示所有的列
            cache:false,                        //设置为 false 禁用 AJAX 数据缓存
            showRefresh: true,                  //是否显示刷新按钮
            minimumCountColumns: 2,             //最少允许的列数
            clickToSelect: true,                //是否启用点击选中行
            showToggle: true,                   //数据显示，切换
            striped: true,                      //各行变色
            sortStable: true,                   //排序
        };

        //将传进来的参数与默认参数做一个合并，相同属性不同值的时候以后者为准
        this.params = $.extend({}, this.defaultSetting, options || {});

        //得到选中的记录
        this.getSelections = function() {
            var arrRes = that.bootstrapTable("getSelections");
            arrRes = removeExtraFields(arrRes);
            return arrRes;
        };

        //得到所有的选中的记录
        this.getAllSelections = function() {
            var arrRes = that.bootstrapTable("getAllSelections");
            arrRes = removeExtraFields(arrRes);
```

2、konckout 自定义绑定组件化。

```
//添加ko自定义绑定
/*element - 涉及此绑定的DOM元素
valueAccessor - 一个JavaScript函数，您可以调用此函数来获取此绑定所涉及的当前模型属性。调用它而不传递任何参数（即调用valueAccessor()）来获取当前的模型属性值。
allBindingsAccessor - 可用于访问绑定到此DOM元素的所有模型值的JavaScript对象
*bindingContext- 保存绑定上下文可用于此元素绑定的对象。
*/
ko.bindingHandlers.bootstrapTable = {
    //绑定时，设置任何初始状态，事件处理程序
    init : function(element, valueAccessor, allBindingsAccessor, viewModel,bindingContext) {
        //这里的oParam就是绑定的viewmodel,valueAccessor()=ko.bootstrapTableViewModel
        var oViewModel = valueAccessor();
        //根据参数执行查询方法
        var $ele = $(element).bootstrapTable(oViewModel.params);

        //劫持方式给oViewModel添加bootstrapTable方法
        oViewModel.bootstrapTable = function() {
            return $ele.bootstrapTable.apply($ele, arguments);
        }
    },
    //绑定之后应用于dom元素上，然后观察dom元素的变化，进行相应调用更新
    update : function(element, valueAccessor, allBindingsAccessor, viewModel,bindingContext) {
    }
};
```

- **knockout.index.js**

1、代码包含传入对象的监控，并将此 js 内的方法，对象全部监控绑定到 KO。

```
        }
        //查询参数监控已经监控绑定
        this.queryCondition = ko.mapping.fromJS(data.queryCondition);
        //自定义下拉数据监控绑定
```

2、默认或者自定义 bootstarptable 参数，并调用 knockout.bootstraptable.js 合并参数，然后进行监控。

```
            } else {
                params.pageNumber = param.pageNumber;
                params.pageSize = param.pageSize;
                params.sortName = data.params.sortName;
                params.sortOrder = data.params.sortOrder;
            }
            return params;
        }
    };

    //将传入的参数与自定义参数进行合并，相同属性不同值的时候以后者为准
    var tableParams = $.extend({}, this.defaultQueryParams,
            data.tableParams || {});
    //绑定表格数据对象
    this.bootstrapTable = new ko.bootstrapTableViewModel(tableParams);
```

3、默认的增删改方法，并提供自定义的方法来扩展。

```
    //新增事件
    this.addClick = function() {
        //申明默认的模态框（此处因为项目中只用到这种模态框，所以未做扩展）
        var dialog = $('<div class="modal" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel"></div>');
        //自定义点击新增前事件
        if (definedMethod.defineAdd != undefined) {
            definedMethod.defineAdd();
        }

        //自定义点击新增传参事件
        if (definedMethod.defineQueryAdd != undefined) {
            //自定义传参事件
            var addQuery = definedMethod.defineQueryAdd();
            //dialog加载页面
            dialog.load(data.urls.add, addQuery, function() {
            });
        } else {
            //加载无参数请求
            dialog.load(data.urls.add, null, function() {
            });
        }
        //body标签下追加dialog
        $("body").append(dialog);
        //dialog的关闭事件
        dialog.modal().on('hidden.bs.modal', function() {
            //关闭弹出框的时候清除绑定（这个清空包括清空绑定和清空注册事件）
            ko.cleanNode(document.getElementById("formEdit"));
            dialog.remove();
            //自定义点击新增后事件
            if (definedMethod.defineAddAfter != undefined) {
                definedMethod.defineAddAfter();
            }
            //刷新table
            self.bootstrapTable.refresh();
        });
    };
```

4、自定义扩展任意方法。

```
    //扩展自定义方法
    this.definedMethod = definedMethod;
```

5、将页面所需的对象，方法 KO 双向绑定并监控。

```
ko.applyBindings(self, bindElement);
```

- **knockout.edit.js**

1、代码包含传入对象的监控，并将此 js 内的方法，对象全部监控绑定到 KO。

```
var that = this;
//修改新增的对象添加监控并绑定
this.editModel = ko.mapping.fromJS(data.editModel);
//如果有下拉对象则监控
if(data.selectData != undefined){
    this.selectData = ko.mapping.fromJS(data.selectData);
}
```

2、调用 bootstrapValidator 进行对象参数的校验，做到实时提示。

```
//默认新增修改配置
this.Default = {
    message: '验证不通过',
    fields: validatorFields,
    //bootstrapValidator的提交方法，验证通过才能提交
    submitHandler: function (validator, form, submitButton) {
        //获取新增编辑的对象并解除监控
```

```
fields: validatorFields,
//bootstrapValidator的提交方法，验证通过才能提交
submitHandler: function (validator, form, submitButton) {
```

```
this.params = $.extend({}, that.Default, {fields: validatorFields} || {});
$('#formEdit').bootstrapValidator(that.params);
```

3、默认提交方法，提供扩展，以及登录前 js 校验。